Exploring the Structure of Data at Scale

Rudy Agovic, PhD
CEO & Chief Data Scientist at Reliancy
January 16, 2019
Outline

• Why exploration of large datasets matters
• Challenges in working with large data sets
• Exploring structure using:
  – K-Nearest Neighbors
  – Clustering
  – Visualization
  – Dimensionality Reduction
  – Topological Data Analysis
• Emphasis on tools and practical usability
Example Use Case

- Modeling Customer Behavior

2 Million visits per day

View
Search
Purchase

500,000 products

- Predict customer intent
- Where do you start?
Example Use Case

• Emerging Topics on Twitter

- 336 Million users
- 350,000 tweets per min
- Changing topics of discussion

• Detect emerging social tensions
• Detect emerging topics of interest
Example Use Case

- Modeling Fraud

- Retail: payment transactions
- Healthcare: insurance charges
- Financial: loan/credit card applications
Frequently Applied Pipeline

- Select how to represent data
- Run a model
- Revise if necessary
- Potential Problems:
  - Fully driven by performance metric (e.g. prediction accuracy)
  - Flying blind in terms of understanding data
  - What if nothing works?
Importance of Exploratory Analysis

• Exploratory Analysis should be a key initial step to drive decisions
• Helps understand the effects of the selected feature representation
• Provides guidance in selection of models
  – What are the properties of the data?
  – What assumptions are made by model?
  – What similarity metrics should I use?
• Reveals structure in data
Challenges in Working with Large Data Sets

- Large number of rows (and columns)
- A lot of popular tools in Python and R
  - Most break once you feed them more than 10,000 data points
  - Most can’t handle large dimensional data
- Tools do exist to work at scale (distributed storage and computation)
- Inherently difficult to understand and explore large and possibly high dimensional data
Understanding Data Structure

Approaches:

– Feature Summary Statistics
– Clustering
– Visualization
– Dimensionality Reduction
– Topological Data Analysis
Understanding Data Structure: Summary Statistics

- Distributions of the individual Features
- Informs about the pre-processing required
- Informs about how they should be combined
• Helps in understanding what potentially relevant “groups” the data holds
• To get the full picture it might not be enough to look at a small subset of the data
Understanding Data Structure: Visualization

- Scientific visualization tools
- Explore and understand what data “looks” like
Understanding Data Structure: Dimensionality Reduction

- Reduce the dimensionality while making sure that “similar” points remain “similar”
- What does similar mean?
- PCA is an example
- Helps in reducing the dimensionality and making problem easier to work with algorithmically
- Helps in visualizing data cloud (2D or 3D)
Understanding Data Structure: Topological Data Analysis

- Represent point cloud in terms of a network of compressed and overlapping shapes
  - circle, sphere, doughnut, loops, holes
- Helps in finding a compressed representation
- Helps in visualizing and exploring data clouds
K-Nearest Neighbors

• Start out with some similarity metric
  – Cosine Similarity
• For each point find k nearest points

• Simple, intuitive algorithm
• Results in graph
• Ability to retrieve similar items
• Moves from data point values to similarities
• Is this really scalable?
• What value does it bring to
  – Clustering
  – Visualization
  – Dimensionality Reduction
Approximate K-Nearest Neighbors

- Scalability addressed by using smart ways to split up search space
- Do not have to look at all other points when searching for k-nearest neighbors
- In K-d trees the search space is repeatedly split
- Many variants exist to speed up retrieval

K-d Tree, repeatedly splits the space
Approximate K-Nearest Neighbors: Annoy

- Developed by Spotify (C++)
- Used in their recommendations pipeline
- Can index large data sets that don’t fit into memory
- Can share the index across CPUs
- Builds a forest of trees for approximation
- Uses random projection to split space within trees
- Open Source
Approximate K-Nearest Neighbors: Faiss

- Developed by Facebook (C++)
- Intended as general purpose tool to model similarity between feature vectors
- Used in internally to search for similar documents (text/images)
- Note: Deep Learning models generate vector representations
- Runs on billion scale data sets
- Provides some clustering capabilities
- Supports GPUs
- Open Source
Approximate K-Nearest Neighbors: nmslib

• Non-Metric Space Library
• Developed by Norwegian PhD students (C++)
• Generic in nature
  – More options
• Comparable in performance
• Easy to use
Using KNN for Clustering

- Step 1: Convert KNN graph to an undirected graph
Using KNN for Clustering

• Step 2: Prune graph. Leave only edges between nodes that share neighbors (Shared Nearest Neighbor Graph)
Using KNN for Clustering

• **Step 3: Assign clusters**
  
  – Sort nodes by the number of edges
  
  – Go through nodes from most connected to least
    
    • If node has not been assigned to cluster add a new cluster and assign it, to node and its neighborhood
    
    • If an encountered node has already been assigned to a cluster, propagate that cluster label to its neighbors
Using KNN for Clustering

• Advantages
  – Simple and intuitive
  – Scales well provided neighborhood graph is computed
  – The number of clusters does not have to be specified
  – The final number of clusters is determined by the structure of the data and the size of the neighborhood chosen when running k-nn and shared nearest neighbors
  – The most dense regions in terms of similarity will be revealed
  – No assumptions are made about the shape or size of clusters
Using KNN for Visualization

• Start out by computing K-Nearest Neighbor graph
• Explore data through visualization
• Example Tool: Gephi
  – Open Source
  – Layout algorithms to arrange nodes visually
  – Most layout algorithms assume some physical force model
    • E.g. edges are like springs, and nodes either attract each other or repel each other
    • The layout algorithm simulates forces and arranges graph visually over multiple iterations
Using KNN for Visualization

- OpenOrd layout algorithm in action
Using KNN for Visualization

- Ability to drill into the data by applying filters
- Ability to color and visualize nodes selectively
Using KNN for Visualization

• Gephi works well for up to 50,000-100,000 nodes
• Useful for getting a deeper understanding of data structure
  – What are the major segments in the data
  – How well is clustering working
  – Where is the model failing
• You would not load several million nodes into it
• It’s not a perfect tool, but it illustrates value
• Graph analysis is an active area of research
• Some commercial alternatives to Gephi:
  – Plantir
  – Polinode
  – Ayasdi
  – Used for visualizing patterns, not necessarily based on KNN
Using KNN in Dimensionality Reduction

- Principle Component Analysis (PCA)

- Project data lower into dimensional space such that as much variance as possible is retained
- Captures linear relationships
Using KNN in Dimensionality Reduction

• First step in most non-linear dimensionality reduction methods: compute KNN

• Capture Relationships using KNN graph
• Reconstruct points in lower dimensional space using the relationships obtained from KNN graph
Using KNN in Dimensionality Reduction

- **LLE (Locally Linear Embedding)**

  1. Select neighbors
  2. Reconstruct with linear weights
  3. Map to embedded coordinates

- Compute Weights to reconstruct each point from its neighbors
- Use weights to find optimal representation in lower dimensional space
Using KNN in Dimensionality Reduction

- **ISOMAP (Isometric Feature Map)**

- Compute Weights based on shortest paths by traversing KNN graph
- Reconstruct data based on those weights in lower dimensional space
Using KNN in Dimensionality Reduction

Why use dimensionality Reduction:
• Using less features helps with computational complexity
• When reduced to 2D or 3D allows us to visualize data

• What about scale?
Neural Networks

- Black box to parametrize functions
Neural Networks

- Variety of activation functions
- Leads to a nested parametrization of functions, similar to what you may remember from calculus

\[ F(x) = A \cdot g(X) + b = A \cdot g(Z \cdot s(x) + c) + b = \ldots \]

Could be hundreds of levels deep
Auto Encoders

- Neural Networks trained to reconstruct inputs
- The hidden layer is used to represent data
Auto Encoders for Dimensionality Reduction at Scale

- Using Tensorflow can run on GPUs
- Learn Representations in lower dimensional space
Encoding Words and Documents

- **Word2Vec Model**
- **Word as input, probability of surrounding words as output**

![Source Text Diagram](image)

<table>
<thead>
<tr>
<th>Source Text</th>
<th>Training Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(the, quick)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(the, brown)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(quick, the)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(quick, brown)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(quick, fox)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(brown, the)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(brown, quick)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(brown, fox)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(brown, jumps)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(fox, quick)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(fox, brown)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(fox, jumps)</td>
</tr>
<tr>
<td>The quick brown fox jumps over the lazy dog.</td>
<td>(fox, over)</td>
</tr>
</tbody>
</table>
Enclosing Words and Documents

- **Word2Vec Model**

![Diagram of Word2Vec Model]

- **Input Vector**: 10,000 positions

- **Hidden Layer Linear Neurons**: 300 neurons

- **Output Layer Softmax Classifier**:
  - **Probability that the word at a randomly chosen, nearby position is “abandon”**
  - **... “ability”**
  - **... “able”**
  - **... “zone”**
Encoding Words and Documents

• Captures contextual relationships
• King – Man + Woman = Queen
• Can be extended to documents (Doc2Vec)
• Gensim – Open source package in Python
Encoding Words and Documents

Bag of Words (Traditional Representation)

- Size of Vocabulary
  - Usually around 20,000-100,000

Word2Vec/Doc2Vec

- Number of Hidden Units
  - Usually set to 100-500
  - Significantly lower dimensionality
Topological Data Analysis

• Mathematical study of shapes (Topology) applied to point clouds

• Strives to capture basic shapes at various resolutions
• Results in compressed and invariant representation of data
  – Can be used as features in Machine Learning
  – Can be used for visualization
• Commercial Vendor: Ayasdi
Topological Data Analysis

- TDA vs. Ayasdi

In TDA:
- Persistent homology (algebraic method) is applied to assign shapes

In Ayasdi’s pipeline:
- Overlapping regions are clustered into nodes
- Connected into a network
Conclusions

- Exploring data at scale is non-trivial
- Exploratory analysis can provide key insights and guidance in modeling choices
- It should be used to guide decisions on
  - Feature selection
  - Model selection
  - Understanding the structure of the data
  - Understanding failure and successes
  - Not flying blind
- Clustering, Visualization, Dimensionality Reduction and Topological Data Analysis are valuable tools in understanding the “shape” of data